Praktische Aufgabe - Classification

In dieser Übung geht es einerseits darum, das erlangte Wissen über gängige Algorithmen für Klassifikationsprobleme praktisch anzuwenden. Andererseits werden dadurch auch Fähigkeiten aus vergangenen praktischen Übungen weiter vertieft, da einige von ihnen wichtige pre-processing Schritte darstellen. Ebenso wie in vorherigen Übungen sind die notwendigen Datensätze für die einzelnen Teilaufgaben online im Materialordner zu finden.

1. Untersuchung verschiedener SVM-Kernels
Die bereits in der Vorlesung kennengelernten Support-Vector-Machines bilden ein mächtiges Werkzeug zum Lösen von Klassifikationsproblemen, besonders unter Einsatz verschiedener Kernels und des so genannten Kernel-Tricks. Um die verschiedenen Kernels näher zu untersuchen, stehen online die Dateien *svm_1.csv*, *svm_2.csv* und *svm_3.csv* zur Verfügung. Gehe wie folgend vor:

a) plotte jeden der Datensätze, wobei Datenpunkte mit unterschiedlichen *class_label* Werten unterschiedlich gefärbt werden

b) nutze SVM mit dem linearen Kernel, dem polynomiellen Kernel und mit einer radialen Basisfunktion und gehe in diesen Schritten vor:
   - probiere für jeden Datensatz systematisch für die verschiedenen Kernels verschiedene Parameter aus und ermittle den erreichten *score*
   - plotte die decision boundaries der trainierten Classifier (siehe Tipps)
   - Für welche Parameter war die Klassifikation besonders gut oder besonders schlecht und an welchen Merkmalen identifizierst du das?
   - interpretiere die Ergebnisse hinsichtlich Anwendbarkeit der Kernels auf verschiedenen Datensätze und verhalten der Kernels mit verschiedenen Parametern
   - welche weiteren Beobachtungen konntest du machen (beispielweise zur Laufzeit)

c) suche dir mindestens einen weiteren Classifier aus und untersuche ihn auf ähnliche Art und Weise (zum Beispiel Decision-Trees liefern häufig interessante decision boundaries für verschiedene Parameter, jedoch kannst du gerne auch völlig neue Classifier nutzen)

Tipps:

- es müssen natürlich je Kernel nur die sinnvollen Parameter überprüft werden, also speziell *degree* für den polynomiellen Kernel und *gamma* für die radiale Basisfunktion
- sehr gute Quellen, um ein besseres Verständnis für Kernels zu entwickeln:

  `http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

  `https://www.quora.com/What-is-the-intuition-behind-Gaussian-kernel-in-SVM`

  `https://charlesmartin14.wordpress.com/2012/02/06/kernels_part_1/`
  (sehr schwer und zäh aber auch sehr informativ)
- gut geeignet, um die decision boundaries darzustellen, ist folgender Code:

```
x_range = numpy.arange(x_start, x_end, x_step)
y_range = numpy.arange(y_start, y_end, y_step)
xx, yy = numpy.meshgrid(x_range, y_range)
# x_step und y_step können angepasst werden für höhere Präzision

grid = numpy.c_[xx.ravel(), yy.ravel()]
# grid hat Form [[0 0] [1 0] ... [x_end y_end]]

z = <Ergebnisse des auf Originaldaten trainierten Classifiers angewendet auf grid>

z = z.reshape(xx.shape)
# notwendige Formatierung für folgende contourf Funktion

matplotlib.pyplot.contourf(xx, yy, z, cmap=plt.cm.RdYlBu, alpha=0.5)
matplotlib.pyplot.show()
```

- zusätzlich können noch die gefärbten Datenpunkte im Diagramm angezeigt werde (was jedoch nicht immer hilfreich ist)

2. Klassifikation mit reellen Daten - der Adult-Datensatz
In dieser Aufgabe geht es darum, den vollständigen Prozess von den Rohdaten bis zur Klassifikation zu durchlaufen. Hierzu stellen wir das Adult Data Set vom UCI Machine Learning Repository (`http://archive.ics.uci.edu/ml/index.html`) zur Verfügung. Ziel ist es anhand anonymisierter Daten über Personen vorherzusagen, ob deren Einkommen über 50.000$ pro Jahr liegt.
Im Materialordner sind hierzu drei Dateien zu finden. Die erste Datei ist *adult_train.csv* und enthält die Daten, die zum trainieren genutzt werden sollen. Die Datei *adult_test.csv* enthält Daten zum vergleichen der Resultate. Ist ein Classifier fertig trainiert, wird er auf diesen Datensatz angesetzt und seine Vorhersagen werden mit den Originalwerten verglichen. Als Gütemaß gilt hier der Prozentsatz korrekt vorhergesagter Werte. Die letzte Datei heißt *adult_info.txt* und beinhaltet nähere Informationen zu dem Datensatz. Es wird folgendes Vorgehen empfohlen:

- verschaffe dir einen Überblick über die Daten (Wertebereiche, Datentypen, missing values)
- behandle beziehungsweise entfernen missing values (siehe Tipps)
- falls nötig konvertiere die Datentypen von Spalten oder kodiere Werte neu (auch hier siehe Tipps)
- ermittle erste Ergebnisse (mindestens mit Decision Tree, Naive Bayes, Nearest Neighbor)
- optimiere die Ergebnisse zum Beispiel durch Feature-Selection, Erzeugung neuer Features und Optimierung der Parameter der Classifier (auch hier siehe Tipps)

Besonders freuen wir uns auf folgende Resultate im Wiki:

- euer bestes Ergebnis für mindestens einen der Classifier und wie es erreicht wurde
- Wurden Datentypen von Features geändert oder Features anders kodiert? Wenn ja, wie?
- Wie wurden missing values behandelt?
- Welche Parameter wurden für den Classifier genutzt und wie wurden sie ermittelt? Vielleicht kannst du sogar einen Plot erstellen, der die accuracy für verschiedene Parameterwerte darstellt.
- Wurden Features generiert oder bestimmte Features selektiert zum Trainieren und Testen? Welche und wie? (gerne auch mit Code Bespielen)

Tipp:

- missing values sind mit einem "?"gekennzeichnet
- es dürfen natürlich auch gerne andere Classifier verwendet werden (es lohnt sich besonders einen Blick auf *Bagging*, *Boosting* und *Random-Forest* zu werfen)
- das *income* sollte am besten auf binäre Werte {0, 1} gemapt werden (siehe hierzu *LabelEncoder*)
- beachte, dass viele Classifier keine Zeichenketten verarbeiten können, ziehe hier besonders die Möglichkeiten *LabelEncoder* und One-Hot-Enconding mit der *pandas.get_dummies()* Funktion in Betracht

3. Classifier evaluieren
Häufig wird die *accuracy* verwendet, um die Qualität eines Classifiers zu testen (wie in Aufgabe 2). Diese ist definiert als

$$\frac{true\_positives+true\_negatives}{true\_positives+true\_negatives+false\_positivs+false\_negatives}$$

wobei *true_positives* die Anzahl der Datenpunkte ist, welche ein positives Klassen-Label haben und auch positiv von dem Classifier klassifiziert wurden. $false\_positives$ bezeichnet hingegen die Datenpunkte, welche positiv klassifiziert wurden obwohl ihr reales Klassen-Label negativ war. Die Begriffe *true_negative* und $false\_negatives$ sind analog definiert.
Dieses Maß ist jedoch nicht immer die beste Wahl, wie wir im folgenden zeigen werden. Hierzu liegt online ein Python-Script bereit, welches zwei Klassen namens *my_classifier* und *my_classifier2* enthält. Diese werden wir nun nutzen, um die *accuracy* mit einem anderen Qualitätsmaß, der so genannten ROC-Kurve und den damit verbundenen Maßen Area-Under-Curve und Youden's J-statistic zu vergleichen. Als theoretische Grundlagen bieten sich folgende Quellen an:

- `https://en.wikipedia.org/wiki/Receiver_operating_characteristic`
- `http://www.dataschool.io/roc-curves-and-auc-explained/` (Video)
- `https://www.svds.com/the-basics-of-classifier-evaluation-part-1/`
- `https://www.svds.com/the-basics-of-classifier-evaluation-part-2/`
- `http://people.inf.elte.hu/kiss/11dwhdm/roc.pdf`

Außerdem greifen wir wieder auf den Datensatz und die trainierten Classifier aus Aufgabe 2 zurück. Gehe zu Beginn so vor:

- trainiere *my_classifier* und *my_classifier2* mit der *fit* Methode wie gewohnt auf den Trainingsdaten (Achtung *income* muss vorher binär kodiert werden, etwa mit *LabelEncoder*)
- vergleiche die *accuracy* mit der deiner Classifier aus Aufgabe 2 anhand der Testdaten
- berechne nun zusätzlich *true-positive-Rate*, *false-positive-Rate*, *true-negative-Rate* und *false-negative-Rate* (jeweils die Anzahl relativ zum zugehörigen original Klassen-Label z.B. $true\_positive\_rate = \frac{true\_positive}{false\_negative+true\_positive}$)

Was fällt dir auf? Wie sind die beiden gegeben Classifier hinsichtlich *accuracy* zu bewerten? Und wie würdest du sie bewerten unter Betrachtung der zusätzlich errechneten Raten? Teile deine Resultate einfach im Wiki.

Um nun diese Ergebnisse mit der ROC-Kurve zu vergleichen, benötigen wir statt exakten Klassen-Labels als Ergebnisse der Vorhersagen unserer Classifier diesmal Wahrscheinlichkeiten für die Zugehörigkeit zu einer Klasse. Dafür verfügen die gegebenen Classifier,

genau wie die meisten Classifier von Sklearn, über eine *predict_proba()* Methode. Wir wollen diese nun so nutzen:

- erstelle und plotte die ROC-Kurve mit der *roc()* Funktion von Sklearn für die gegebenen und deine eigenen Classifier (anhand der Testdaten)
- nutze dazu die Wahrscheinlichkeiten für das Klassen-Label 1, welche du mit der *predict_proba()* Methode ermitteln kannst (genauere Infos hierzu sind in der Dokumentation)
- berechne jeweils den zugehörige Area-Under-Curve Score mit der Funktion *auc()*
- berechne den Maximalwert der Youden's J-statistic

Inwiefern verhalten sich diese Qualitätsmaße anders als die *accuracy*? Und wie ist dies zu bewerten? Kannst du eine Regelung finden, wann der Area-Under-Curve Score eine hohes Ergebnis liefert? Und wie verhalten sich die Werte der Youden's J-statistic? Eventuell hilft es zur Interpretation für jeden Classifier je original Klassen-Label ein Histogramm der vorhergesagten Wahrscheinlichkeiten für das Klassen-Label 1 zu plotten. Auch hier sind wieder alle Ergebnisse im Wiki willkommen.

Zusatzaufgabe: Klassifikation mit reellen Daten - Data Mining Cup
Auch für diese Aufgabe liegen eine Trainingsdatei, genannt *dmc_train.csv* und eine Testdatei, genannt *dmc_test.csv* im Materialordner bereit. Es handelt sich hierbei um eine weitestgehend bereinigte Teilmenge der Data Mining Cup Daten. Das Ziel *returnQuantity* wurde hier in eine binäre Variable umgewandelt (0 oder >0).
Ziel ist es eine möglichst präzise Vorhersage der Testdaten zu schaffen, gemessen analog zur Aufgabe 2. Dabei dürfen nach Belieben Techniken, wie etwa das Erzeugen neuer Features, Feature-Selection und Neu-Kodierung von Attributen eingesetzt werden. Weiterhin dürfen wie in Aufgabe 2 natürlich beliebige Classifier verwendet werden, auch solche, die nicht in der Vorlesung vor kamen.
Hochzuladen sind das beste erreichte Ergebnis samt grober Skizze wie dieses erreicht wurde (auch hier siehe Aufgabe 2).